ARQUITECTURA DE TIEMPO REAL PARA EL CONTROL DE ACTITUD EN UN SATÉLITE EXPERIMENTAL*

Juan Zamorano, Ángel Esquinas, Peter Bradley, Daniel Brosnan,
Jorge Garrido, Emilio Salazar, Alejandro Alonso, Juan A. de la Puente

Grupo de Sistemas de Tiempo Real y Arquitectura de Servicios Telemáticos (STRAST)
Universidad Politécnica de Madrid (UPM)

Resumen

La misión UPMSat-2 tiene por objetivo la construcción y el lanzamiento de un microsatélite experimental que sirva como plataforma de educación e investigación en diversos aspectos de la ingeniería de sistemas espaciales. En este artículo se describe la arquitectura y el diseño del software de tiempo real que realiza el control de actitud del satélite. La arquitectura está basada en modelos, y las herramientas utilizadas permiten la validación de las propiedades de tiempo real y la generación de código de forma casi totalmente automática.

Palabras clave: Sistemas de control por computador, sistemas de tiempo real, diseño de software basado en modelos.

1. INTRODUCCIÓN

UPMSat-2 es un proyecto de la Universidad Politécnica de Madrid (UPM) que tiene como objetivo la construcción, puesta en órbita y operación de un satélite experimental que sirva como demostrador tecnológico y plataforma educativa en el ámbito de los sistemas espaciales. El proyecto está liderado por el Instituto Ignacio Da Riva(IDR/UPM), y se lleva a cabo con la colaboración del grupo STRAST/UPM,la empresa TECNOBIT,y otros grupos universitarios y empresas.

UPMSat-2 es un microsatélite de unos 50 kg de masa, con una envolvente geométrica de $0,5 \times 0,5 \times 0,6$ m. La órbita prevista es polar heliosíncrona a unos 600 km de altura. El lanzamiento está previsto para 2015, y la vida útil de la misión se estima en unos dos años.

El grupo de Sistemas de Tiempo Real y Arquitectura de Servicios Telemáticos (STRAST) se ocupa de desarrollar el software para el computador embarcado y la estación de tierra de la misión. En este artículo se describen los aspectos relacionados con el software de tiempo real que efectúa el control de actitud del satélite. En el apartado 2 se resumen las principales características de la arquitectura de hardware y software del computador embarcado. Los elementos principales del sistema de control de actitud se describen en el apartado 3, y el software de tiempo real correspondiente se describe en el apartado 4. Por último, el apartado 5 resume las principales conclusiones del trabajo.

*Trabajo financiado parcialmente por el Plan Nacional de I+D+I, proyecto TIN2011-28567-C03-01 (HI-PARTES).

2. COMPUTADOR EMBARCADO

El computador embarcado (OBC, *on-board computer*) del satélite realiza las siguientes funciones:

- Control de actitud (ADCS, *Attitude determination and control)*;
- supervisión de la plataforma del satélite (*housekeeping*);
- gestión de mensajes de telemando (TC, *telecommand*) y telemedida (TM, *telemetry*);
- gestión de los experimentos que constituyen la carga útil del satélite.

La figura 1 muestra las interfaces del computador con los sensores de que se dispone para realizar estas funciones.

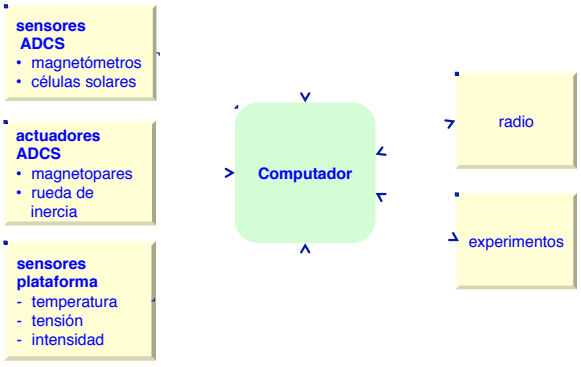

Figura 1: Diagrama de contexto del computador embarcado.

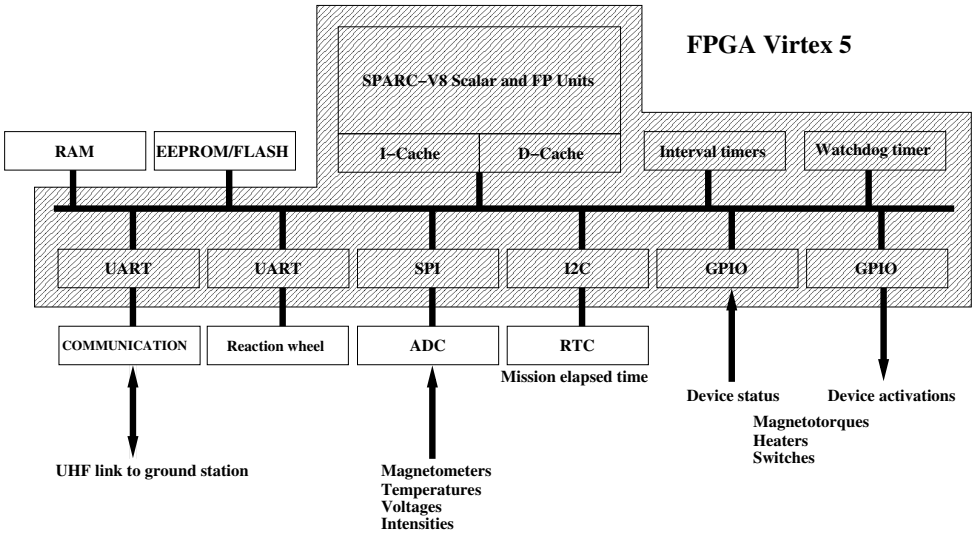

Figura 2: Computador embarcado.

2.1. ARQUITECTURA DE HARDWARE

El hardware del computador embarcado está basado en un procesador LEON3 con arquitectura SPARC v8 [8, 14], con memoria RAM y memoria no volátil tipo EEPROM para almacenar los datos de configuración y de telemedida para su posterior envío a la estación de tierra. El computador incluye también interfaces de entrada y salida analógica y digital para la interacción con los sensores, así como interfaces RS-232 y RS-422 para la radio, las conexiones de depuración y pruebas, y algunos de los experimentos [4].

La versión de ingeniería de la OBC se ha desarrollando a partir de la biblioteca de *IP cores* en VHDL GRLIB. Esta biblioteca contiene el modelo sintetizable del procesador de 32 bits LEON3 junto con controladores de buses de sistema AMBA, controladores de memoria así como controladores de buses auxiliares y dispositivos periféricos. La biblioteca está disponible en código fuente VHDL y se distribuye con licencia GNU GPL. Por lo tanto, es posible desarrollar (SOC, *Systems On a Chip*) para su uso en computadores de a bordo con una adecuada configuración y síntesis sobre dispositivos programables como FPGAs.

La figura 2 muestra la estructura del computador embarcado que se ha sintetizado en una FPGA, la versión de ingeniería del computador usa una tarjeta de evaluación ML507 de la FPGA de Xilinx Virtex-5 FXT. Esta versión de ingeniería se usa para el desarrollo del software embarcado y sirve de base para la versión de vuelo del computador que contendrá *chips* de memoria y FPGA resistentes a la radiación. El coste de estos componentes y los requisitos de consumo de energía limitan la cantidad de memoria disponible a 4 MB de SRAM y 1 MB de EEPROM. Por motivos similares, la velocidad del procesador LEON3 es de 50 MHz.

2.2. ARQUITECTURA DE SOFTWARE

El software del computador embarcado se ha diseñado siguiendo un enfoque basado en modelos (MDE) [1]. Los distintos subsistemas se modelan en lenguajes de alto nivel como Simulink® o el perfil MARTE de UML [11, 12]. A partir de los modelos de alto nivel se genera automáticamente el código fuente en C o Ada utilizando herramientas asociadas a los lenguajes de modelado, junto con otras desarrolladas por el grupo [13].

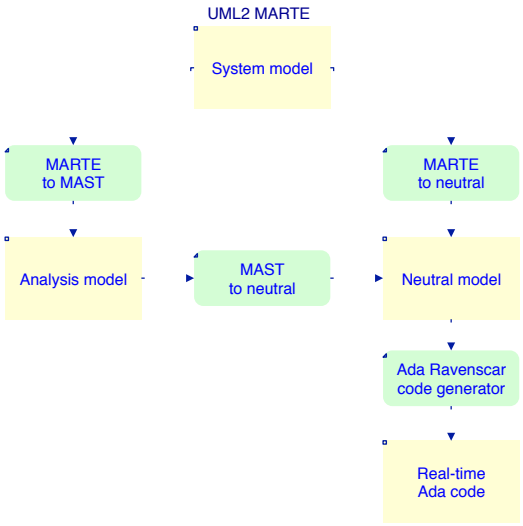

Figura 3: Proceso de desarrollo de software.

La figura 3 muestra de forma esquemática el proceso de desarrollo utilizado para el software.

El software embarcado tiene requisitos de tiempo real y se hace necesario la realización de un análisis de planificabilidad para verificar que las acciones de los subsistemas se realizan en el momento correcto [6]. Por lo tanto, las herramientas incluyen restricciones para asegurar que los modelos sean compatibles con el perfil de Ravenscar [3]. El perfil de Ravenscar se definió como un conjunto de restricciones a la parte concurrente del lenguaje de programación Ada, con el objetivo que la arquitectura de software resultante se pueda analizar temporalmente. Aunque el perfil se definió originalmente para el lenguaje Ada, su modelo computacional se puede extrapolar a otros paradigmas de programación concurrente como POSIX. La plataforma de ejecución está basada en el núcleo de tiempo real ORK+ [5], que da soporte al perfil de Ravenscar. Así que tanto el compilador como el núcleo de tiempo real realizan comprobaciones del código generado para verificar que se cumplan las restricciones.

3. CONTROL DE ACTITUD

La actitud del satélite está representada por la orientación de los ejes del satélite con respecto a un sistema de referencia local vertical. Esta orientación viene dada por la matriz de transformación entre ambos sistemas de referencia, construida a partir de los ángulos de Euler, ϕ, θ y ψ. El objetivo del sistema de control de actitud es mantener el eje Z del satélite orientado hacia el centro de gravedad de la Tierra, mientras que su eje Y se orienta en una dirección normal al plano de la órbita. El satélite debe girar entorno al eje X con una velocidad angular de $0,01\,\mathrm{rad/s}$ con el objetivo de mantener una temperatura homogénea en las caras del satélite.

La actitud del satélite se puede determinar a partir de las medidas obtenidas por tres magnetómetros ortogonales, que permiten determinar el valor y la dirección del campo magnético terrestre en el sistema de referencia local del satélite. Para modificar la actitud del satélite se dispone de tres magnetopares, que generan un momento de rotación al interaccionar con el campo magnético terrestre.

Para diseñar el algoritmo de control se ha simulado la dinámica del satélite y las fuerzas externas que actúan sobre él mediante un modelo en Simulink (figura 4).

Las variables del modelo representan los distintos momentos de fuerzas que actúan sobre el satélite, los cuaterniones que representan la actitud del satélite respecto al sistema de referencia vertical, la matriz de rotación desde el sistema de ejes del satélite ($\mathbf{C}_{BV}$), y los componentes de la velocidad

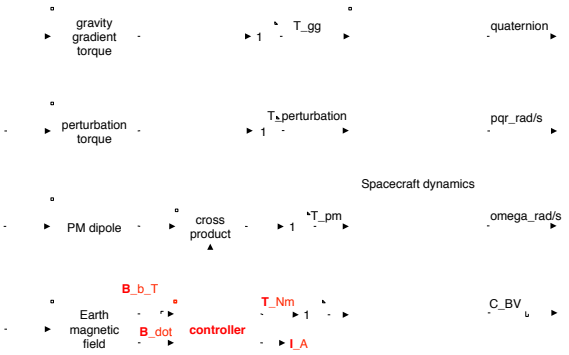

Figura 4: Modelo de la dinámica del satélite.

angular (**pqr**).

El bloque *controller* representa el algoritmo de control, que tiene como entradas las medidas del campo magnético proporcionadas por los magnetómetros ($\mathbf{B}_{b_T}$) y sus derivadas en el tiempo ($\mathbf{B}_{\mathrm{dot}}$). Las salidas del controlador son las intensidades que se suministran a los magnetopares ($\mathbf{I}_A$) y los pares que ejercen éstos ($\mathbf{T}_{\mathrm{Nm}}$). Los sensores y actuadores propiamente dichos están incluidos en el bloque controlador.

Los detalles del algoritmo y su justificación pueden verse en el documento [7]. El periodo de muestreo del algoritmo de control es de un segundo. La adquisición de los datos de los magnetómetros se realiza mediante el conversor analógico digital del OBC (figura 2) con una resolución de 12-bits y la actuación sobre los magnetopares se hace mediante modulación de ancho de impulso (PWM, *Pulse Width Modulation*) mediante salidas digitales del OBC.

4. SOFTWARE DE CONTROL

El software del sistema de control de actitud (ADCS) tiene varios componentes, que se representan en el diagrama MARTE-UML de la figura 5.

El componente AttitudeControl se ejecuta periódicamente e invoca la función ControlAlgorithm, cuyo código fuente en C se genera automáticamente a partir del modelo de simulación anterior. Los demás componentes permiten ajustar los parámetros del controlador, proporcionar medidas al sistema de telemetría, y ajustar el controlador en función de los cambios de modos del sistema. Estos componentes se deben ejecutar concurrentemente con otros subsistemas de software.

El entorno de desarrollo dispone de herramientas para calcular el peor caso de tiempo de cómputo (WCET, *Worst Case Execution Time*) [2] de

Figura 5: Estructura del software del sistema ADCS.

las funciones del sistema. Entre estas funciones está el control de actitud, los detalles del cálculo del WCET pueden verse en el documento [9]. Con la arquitectura de software y los WCET de las funciones del sistema se analizan los requisitos de tiempo real mediante herramientas específicas [10].

Una vez comprobado el comportamiento temporal, se genera todo el esqueleto del código concurrente y de tiempo real en Ada a partir del modelo MARTE-UML [13]. La compilación e implantación en el computador embarcado se realizan de forma inmediata usando la cadena de compilación GNAT/ORK [15].

5. CONCLUSIONES

El satélite UPMSat-2 (figura 6)es una plataforma de gran interés para la experimentación de nuevas técnicas por parte de los grupos de investigación de la UPM.

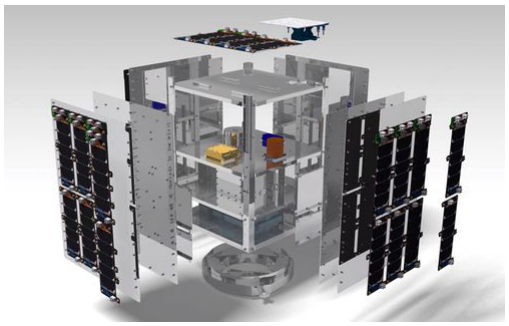

Figura 6: Vista general del satélite UPMSat-2.

En particular, las técnicas de desarrollo de software de tiempo real en las que trabaja el grupo STRAST se han aplicado al diseño del software de control del satélite. Los resultados obtenidos hasta ahora son satisfactorios, y por tanto está previsto continuar con el desarrollo del resto del software embarcado utilizando estas mismas técnicas en los próximos meses.

Agradecimientos

Los autores agradecen la colaboración del resto del equipo del proyecto UPMSat2, especialmente la ayuda prestada pro Assal Farrahi, Javier Cubas y Ángel Sanz.

Referencias

[1] Alejandro Alonso, Emilio Salazar, and Juan A. de la Puente. Design of on-board software for an experimental satellite. In *Jornadas de Tiempo Real — JTR-2013*, 2103. Available at `www.dit.upm.es/~str/papers/pdf/alonso&13a.pdf`.

[2] Esteban Asensio, Jorge López, Juan A. de la Puente, and Juan Zamorano. Herramientas de análisis temporal para el desarrollo de sistemas de tiempo real críticos. In *Proc. CEDI 2010*, September 2010. (In Spanish). Submitted for publication.

[3] Alan Burns, Brian Dobbing, and Tullio Vardanega. Guide for the use of the Ada Ravenscar profile in high integrity systems. Technical Report YCS-2003-348, University of York, 2003.

[4] Juan A. de la Puente, Juan Zamorano, Alejandro Alonso, and Daniel Brosnan. A real-time computer control platform for an experimental satellite. In *Jornadas de Tiempo Real — JTR-2012*, 2012. Available at `http://www.ctr.unican.es/jtr12/programme.html`.

[5] Juan A. de la Puente, Juan Zamorano, José A. Pulido, and Santiago Urueña. The ASSERT Virtual Machine: A predictable platform for real-time systems. In Myung Jin Chung and Pradeep Misra, editors, *Proceedings of the 17th IFAC World Congress*. IFAC-PapersOnLine, 2008.

[6] European Cooperation for Space Standardization. *ECSS-E-ST-40C Space engineering — Software*, March 2009. Available from ESA.

[7] Assal Farrahi, Javier Cubas, and Ángel Sanz. Design of the attitude control subsystem of the upmsat-2 satellite. Technical report, Instituto de Microgravedad Ignacio da Riva, 2013.

[8] Gaisler Research. *LEON3 - High-performance SPARC V8 32-bit Processor. GRLIB IP Core User's Manual*, 2012.

[9] Jorge Garrido, Daniel Brosnan, Juan A. de la Puente, Alejandro Alonso, and Juan Zamorano. Analysis of WCET in an experimental satellite software development. In Tullio Vardanega, editor, *12th International Workshop on Worst-Case Execution Time Analysis*, volume 23 of *OpenAccess Series in Informatics (OASIcs)*, pages 81–90. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012.

[10] Michael González Harbour, J. Javier Gutiérrez, J. Carlos Palencia, and José María Drake. MAST modeling and analysis suite for real time applications. In *Proceedings of 13th Euromicro Conference on Real-Time Systems*, pages 125–134, Delft, The Netherlands, June 2001. IEEE Computer Society Press.

[11] *OMG Unified Modeling Language (UML)*, 2011. Version 2.4.1.

[12] *OMG UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems*, 2011. Version 1.1.

[13] Emilio Salazar, Alejandro Alonso, Miguel A. de Miguel, and Juan A. de la Puente. A model-based framework for developing real-time safety Ada systems. In *Reliable Software Technologies - Ada-Europe 2013*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013.

[14] SPARC International, Upper Saddle River, NJ, USA. *The SPARC architecture manual: Version 8*, 1992.

[15] Juan Zamorano and José F. Ruiz. GNAT/ORK: An open cross-development environment for embedded Ravenscar-Ada software. In Eduardo F. Camacho, Luis Basañez, and Juan Antonio de la Puente, editors, *Proceedings of the 15th IFAC World Congress*. Elsevier Press, 2003.